Gaming Behavior Analysis

The objective

The primary goal of this project is to **decode gaming behavior** by analyzing player and level data using SQL queries.

In this project,I work with dataset related to a gaming, The dataset includes two tables:
`Player Details` and `Level Details`.

By :FOLAYEMI O.AKINBOBOBLA

Technical Skills Applied

1. **Database Management**:
   - I created a database named **“Gaming Behavior Analysis/Mentorness ,**Managing databases involves setting up, maintaining, and optimizing data storage for efficient retrieval.
2. **SQL Querying**:
   - SQL (Structured Query Language) is the backbone of your analysis.
   - I wrote various SQL queries to extract, transform, and analyze data.
   - Specific query techniques include:
     - **Joining Tables**: Combining data from different tables using JOIN.
     - **Aggregation and Grouping**: Using GROUP BY to aggregate data (e.g., calculating average kill count).
     - **Window Functions**: Utilizing functions like ROW_NUMBER() and RANK() for ranking and calculations.
     - **Subqueries**: Incorporating subqueries to retrieve intermediate results.
     - **Order By**: Sorting results using ORDER BY.
     - **Aggregate Functions**: Performing calculations on grouped data (e.g., SUM, MIN).
3. **Data Cleaning and Transformation**:
   - I cleaned data by removing unnamed or unwanted columns.
   - Standardized date names using Google Sheets.
   - Data transformation ensures data consistency and prepares it for analysis.
   - Ensuring data integrity prevents inconsistencies and improves reliability.

4. **Problem-Solving and Analysis**:

- I addressed specific problem statements related to gaming behavior.
- Analyzing kill counts, lives earned, and player interactions led to valuable insights.

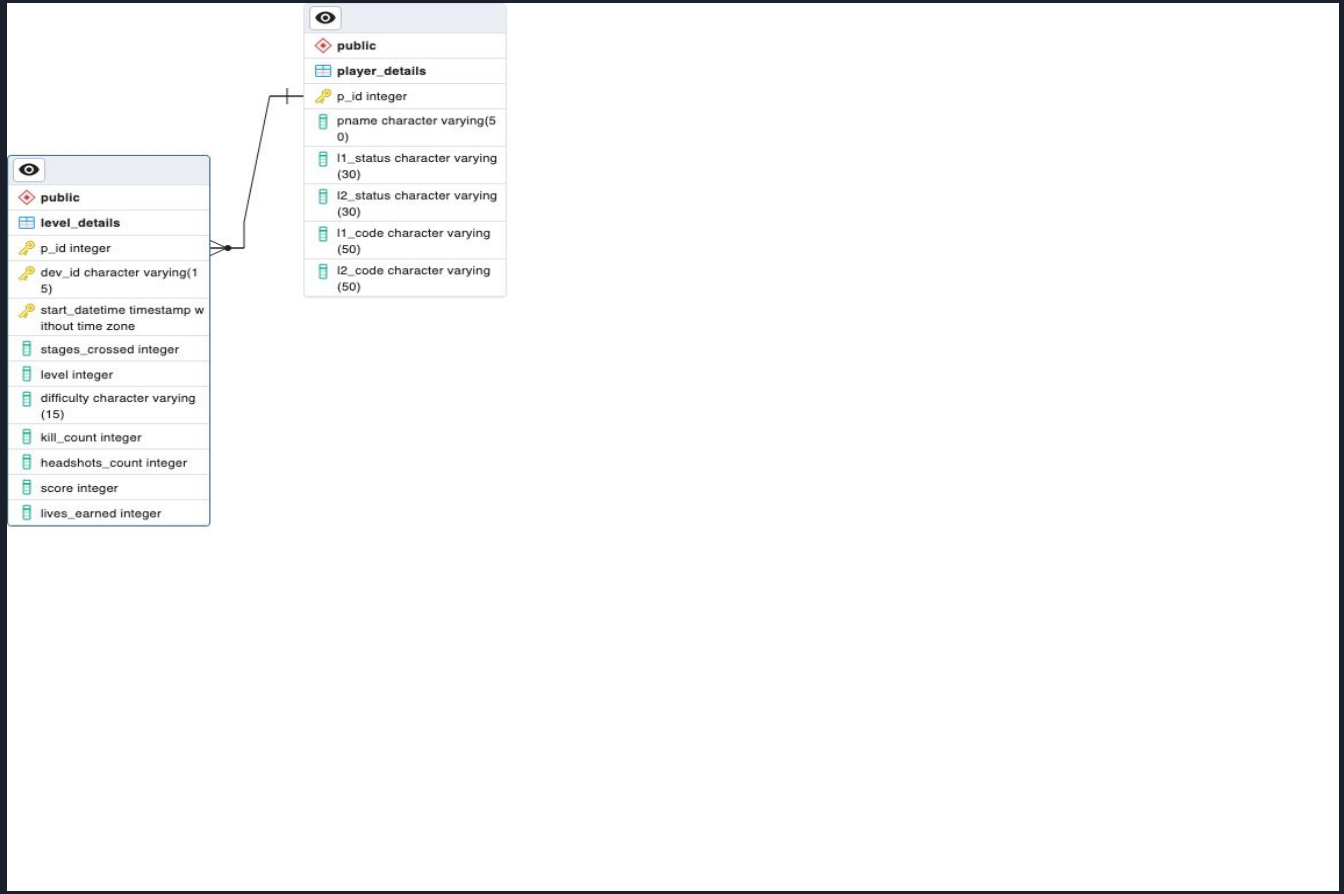
The queries and insight to the problem statement can be found

HERE

:https://github.com/Follyemzy/Gaming-Behaviour-Analysis-https://github.com/Follyemzy/Gaming-Behaviour-Analysis-

SEE SCHEMERS BELOW

SCHEMERS


public
player_details
p_id integer
pname character varying(50)
l1_status character varying (30)
l2_status character varying (30)
l1_code character varying (50)
l2_code character varying (50)
public
level_details
p_id integer
dev_id character varying(15)
start_datetime timestamp without time zone
stages_crossed integer
level integer
difficulty character varying (15)
kill_count integer
headshots_count integer
score integer
lives_earned integer

Query Solutions

```sql
alter table level_details add primary key(P_ID,Dev_id,start_datetime);

-- 1. Extract `p_id`, `Dev_ID`, `PName`, and `difficulty_level` of all players at Level 0.

select ld.p_id,ld.dev_id,pd.pname,ld.difficulty,ld.level
from level_details ld
join player_details pd
on ld.p_id = pd.p_id
where ld.level=0;

-- 2. Find `Level1_code`wise average `Kill_Count` where `lives_earned` is 2, and at least 3 stages are crossed.

select pd.l1_code,ld.lives_earned,ld.stages_crossed,
ROUND(avg(ld.kill_count))as avg_kill_count
from level_details ld
join player_details pd
on ld.p_id = pd.p_id
where ld.lives_earned=2
and ld.stages_crossed>=3
group by pd.l1_code,ld.lives_earned,ld.stages_crossed;

-- 3. Find the total number of stages crossed at each difficulty level for Level 2, with players
--using `zm_series` devices. Arrange the result in decreasing order of the total number of stages crossed.

select ld.dev_id,ld.difficulty,ld.level,
sum(ld.stages_crossed)as Total_stagesCrossed
from level_details ld
 join player_details pd
on ld.p_id = pd.p_id
where ld.level = 2
and ld.dev_id like 'zm%'
group by ld.dev_id,ld.difficulty,ld.level,pd.l2_status
order by Total_stagesCrossed desc;

-- 4. Extract `P_ID` and the total number of unique dates for those players who have played games on multiple days.

select pd.pname, pd.p_id,count(distinct ld.start_datetime) as date_count
from player_details pd
 join level_details ld
on ld.p_id = pd.p_id
group by pd.p_id,pd.pname
having count(distinct ld.start_datetime)>1
order by date_count desc;
```

```sql
-- 5. Find `P_ID` and levelwise sum of `kill_counts` where `kill_count` is greater than the
-- average kill count for Medium difficulty.

select ld.p_id, ld.level, sum(ld.kill_count) as total_killcount
from level_details ld
join (
    select p_id, avg(kill_count) as avg_kill_count
   from level_details
  where difficulty = 'Medium'
    group by p_id) as avg_kills on ld.p_id = avg_kills.p_id
where ld.kill_count > avg_kills.avg_kill_count
group by ld.p_id, ld.level;


-- 6. Find `Level` and its corresponding `Level_code`wise sum of lives earned, excluding Level0.
--Arrange in ascending order of level.

select l1_code,l2_code,ld.level,
sum(lives_earned)as totalLives_earned
from player_details pd
join level_details ld
on pd.p_id = ld.p_id
where level >0
group by ld.level,l1_code,l2_code
order by ld.level asc;

-- 7. Find the top 3 scores based on each `Dev_ID` and rank them in increasing order using
-- `Row_Number`. Display the difficulty as well.

select dev_id, score, difficulty, rank
from (select dev_id,score, difficulty,
  row_number()over(partition by dev_id order by score asc) as rank
  from level_details)
as Ranked_scores
where rank <= 3
order by dev_id, rank;

-- 8. Find the `first_login` datetime for each device ID.
select dev_id,min(start_datetime) as first_login
from level_details
group by dev_id;
```

```sql
-- 9. Find the top 5 scores based on each difficulty level and rank them in increasing order
-- using `Rank`. Display `Dev_ID` as well.

select dev_id,score,difficulty, rank
from ( select dev_id,score,difficulty,
  rank() over (partition by difficulty order by score asc) as rank
  from  level_details) AS RankedScores
where rank <= 5
order by  difficulty,rank,dev_id;

-- 10. Find the device ID that is first logged in (based on `start_datetime`) for each player
-- (`P_ID`). Output should contain player ID, device ID, and first login datetime.

select pd.p_id,ld.dev_id,
min(ld.start_datetime) as first_logindatetime
from level_details ld
join player_details pd
on ld.p_id=pd.p_id
group by ld.dev_id,pd.p_id;

-- 11. For each player and date, determine how many `kill_counts` were played by the player so far.
-- a) Using window functions
-- b) Without window functions

a) select pd.pname,start_datetime,kill_count,
sum(kill_count)over(partition by start_datetime)as "Killcount_soFAr"
from level_details ld
join player_details pd
on ld.p_id=pd.p_id
order by  pd.pname,start_datetime;


-- 12. Find the cumulative sum of stages crossed over `start_datetime` for each `P_ID`,
-- excluding the most recent `start_datetime`.
select * from level_details

select ld.P_ID, ld.start_datetime, ld.stages_crossed,
(select sum(ld2.stages_crossed)
from level_details ld2
where ld2.P_ID = ld.P_ID and ld2.start_datetime < ld.start_datetime)
as cumulative_stages
from level_details ld
order by ld.P_ID, ld.start_datetime;
```

```sql

-- 13. Extract the top 3 highest sums of scores for each `Dev_ID` and the corresponding `P_ID`.

with RankedScores as (
select dev_id, p_id, sum(score) as total_score,
row_number()over (partition by dev_id order by sum(score) desc)as rn
from level_details
group by dev_id, p_id)
select dev_id, p_id, total_score
from RankedScores
where rn <= 3
order by dev_id, total_score desc

-- 14. Find players who scored more than 50% of the average score, scored by the sum of scores for each `P_ID`.

select pd.pname,pd.p_id,round(avg(score))
from player_details pd
join level_details ld
on ld.p_id=pd.p_id
group by pd.pname,pd.p_id
having round(avg(score))>0.5 *
    (select sum(score)
    from level_details
     where p_id=pd.p_id);

15)-- Create a stored procedure to find the top `n` `headshots_count` based on each `Dev_ID`
--and rank them in increasing order using `Row_Number`. Display the difficulty as well.

create or replace function  GetTopNHeadshots(n int)
    returns table (dev_id int, headshots_count int, difficulty varchar)as $$ begin
return query
 with RankedHeadshots as(
select  dev_id,headshots_count, difficulty,dev_id,headshots_count,difficulty,
        row_number() over
   (partition by dev_id order by headshots_count asc) as RowNum
        from level_details )
  select dev_id, headshots_count, difficulty
  from RankedHeadshots
   where RowNum <= n;
end;
$$ language plpgsql;
```

Launchpad

Thank yo